Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Intrusion detection using rough set classification*

ZHANG Lian-hua (张连华)[†1,2], ZHANG Guan-hua (张冠华)[1], YU Lang (郁 郎)[3],
ZHANG Jie (张 洁)[4], BAI Ying-cai (白英彩)[1]

([1]*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200030, China*)
([2]*An Zong Information Technology Inc., Shanghai 200042, China*)
([3]*www.antpower.org*)
([4]*Department of Computing, Hong Kong Polytechnic University, Hong Kong, China*)
[†]E-mail: a000309035@21cn.com

Received May 21, 2003; revision accepted July 21, 2003

Abstract: Recently machine learning-based intrusion detection approaches have been subjected to extensive researches because they can detect both misuse and anomaly. In this paper, rough set classification (RSC), a modern learning algorithm, is used to rank the features extracted for detecting intrusions and generate intrusion detection models. Feature ranking is a very critical step when building the model. RSC performs feature ranking before generating rules, and converts the feature ranking to minimal hitting set problem addressed by using genetic algorithm (GA). This is done in classical approaches using Support Vector Machine (SVM) by executing many iterations, each of which removes one useless feature. Compared with those methods, our method can avoid many iterations. In addition, a hybrid genetic algorithm is proposed to increase the convergence speed and decrease the training time of RSC. The models generated by RSC take the form of "IF-THEN" rules, which have the advantage of explication. Tests and comparison of RSC with SVM on DARPA benchmark data showed that for Probe and DoS attacks both RSC and SVM yielded highly accurate results (greater than 99% accuracy on testing set).

Key words: Intrusion detection, Rough set classification, Support vector machine, Genetic algorithm
doi:10.1631/jzus.2004.1076 **Document code:** A **CLC number:** TP393

INTRODUCTION

Intrusion detection is used to classify normal and intrusive activities, in which machine learning can play an important role. Recently the machine learning-based intrusion detection approaches (Allen *et al.*, 2000) have been subjected to extensive researches because they can detect both misuse and anomaly. The learning-based intrusion detection approaches include two key steps: feature extraction and detection model generation. In the research of feature extraction in intrusion detection, Wenke (1999) used improved Apriori algorithm to acquire features of network connection level. This method is very effective. Later, Srinivas and Sung (2002) presented the use of support vector machine (SVM) to rank these extracted features, but this method needs many iterations and is very time-consuming. In the research of detection model generation, it is desirable that the detection model be explainable and have high detection rate, but the existing methods cannot achieve these two goals. For example, neural networks (James, 1998) could achieve high detection rate but the detection rules

* Project (No. 2001AA40437.2) partially supported by the Hi-Tech Research and Development Program (863) of China

generated are not explainable; decision trees (Wenke, 1999) could yield explainable rules but the detection rate is low.

In this paper we present the use of rough set classification (RSC) (Pawlak, 1982) for intrusion detection system (IDS) feature ranking and intrusion detection rules generation. Intrusion detection using RSC can yield both explainable detection rules and high detection rate for some attacks, and feature ranking using RSC for IDS is simple and fast.

RSC is one of the important contents of rough set theory (Wang and Tao, 2003). The main contribution of rough set to learning theory is the concept of reducts. A reduct is a minimal subset of attributes with the same capability of objects classification as the whole set of attributes. In this paper, we propose a fast hybrid genetic algorithm for the reduct computation of rough set. In fact, the reduct computation of rough set corresponds to feature ranking for IDS in RSC. Compared with the classic SVM based feature ranking approach (Srinivas and Sung, 2002), this feature ranking method is simpler and faster.

RSC creates the intrusion (decision) rules using the reducts as templates. After reduct generation, the detection rules are automatically computed subsequently. The rules generated have the intuitive "IF-THEN" format, which is explainable and very valuable for improving detector design. Experiments were designed to test the rules detection performance. The experiment data we used originated from MIT's Lincoln Labs. It was developed for KDD (1999) competition by DARPA and is considered a standard benchmark for intrusion detection evaluations. Since SVM performed well among the classical intrusion detection algorithms (Srinivas and Sung, 2002), we also use SVM to detect intrusions on the same dataset for comparison. The test results indicated that RSC algorithm has compatible level detection performance with SVM algorithm for detection of Probe and DoS attacks (all above 99%) on DARPA dataset. But RSC has obvious advantage in rules explanations. Further comparisons between RSC based IDS and SVM based IDS are provided in detail in the paper.

The paper is organized as follows. In the second section, the system model for general machine learning-based intrusion detection approach is introduced. In the third section, rough set is preliminarily interpreted and rough set classification algorithm used in this paper and the hybrid genetic algorithms proposed are explained in detail. Experiment design for comparison of RSC based IDS and SVM based IDS is given in the fourth section to indicate the advantages of RSC algorithm. Finally we conclude the paper in the last section.

SYSTEM MODEL

Based on the research work of Wenke (1999), designing an intrusion detection system based on learning algorithm can be described in the following steps:

(1). Capture network data by using tools such as Tcpdump, Dsniff, etc.; (2). Process these data into suitable input format; (3). Normalize the network flow and extract features of attack behavior or normal usage pattern from raw data; (4). Design and use learning algorithm to get detection rules; (5). Integrate the detection rules into the real time IDS for detecting intrusion.

In these five steps, as the above section said, feature extraction and detection rules generation are two key steps. For feature extraction, it depends on data source and the category of attack to be detected. In order to focus on our learning algorithm study, we choose the 1999 KDD intrusion detection contest dataset to design our system. The 1999 KDD intrusion detection contest used 1998 DARPA intrusion detection dataset to construct the connection records and extract the object features (Wenke, 1999). 1998 DARPA intrusion detection dataset was acquired from nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN and peppered with four main categories of attacks: DoS, Probe, U2R, R2L. A connection record is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is

labeled as either normal, or as an attack, with exactly one specific attack type. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted. The following three main feature sets can be used to classify each connection.

1) Intrinsic features, i.e., general information related to the connection. They include the duration, type, protocol, flag, etc. of the connection;

2) Traffic feature, i.e., statistics related to past connections similar to the current one, e.g., number of connections with the same destination host or connections related to the same service in a given time window or within a predefined number of past connections;

3) Content features, i.e., features containing information about the data content of packets ("payload") that could be relevant to discover an intrusion, e.g., errors reported by the operating system, root access attempts, etc.

For detection rules auto-generation, we present the use of rough set classification for this task. It includes three phases:

1) Preprocessing: The raw data is first partitioned into three groups: DoS attack detection dataset, Probe attack detection dataset, U2R&R2L attack detection dataset. For each dataset, a decision system is constructed. Each decision system is subsequently split into two parts: the training dataset and the testing dataset.

2) Training: rough set classifier is trained on each training dataset of three different types of attacks (DoS, Probe, U2R&R2L). Each training dataset uses the corresponding input features and fall into two classes: normal (+1) and attack (−1).

3) Testing: measure the performance on testing data.

We will describe our rough set classification algorithm in detail in the following section.

ROUGH SET CLASSIFICATION ALGORITHM

Part A: Rough set theory preliminary

Rough sets theory was developed by Zdzislaw Pawlak in the early 1980's (Pawlak, 1982). It is a mathematical tool for approximate reasoning for decision support and is particularly well suited for classification of objects. Rough sets can also be used for feature selection, feature extraction etc. (Wang, 2001).

Definition 1 An information system is defined as a four-tuple as follows, $S=<U, Q, V, f>$, where $U=\{x_1, x_2, \ldots, x_n\}$ is a finite set of objects (n is the number of objects); Q is a finite set of attributes, $Q=\{q_1, q_2, \ldots, q_n\}$; $V=\bigcup_{q\in Q} V_q$ and V_q is a domain of attribute q; $f: U\times V\to V$ is a total function such that $f(x, q)\in V_q$ for each $q\in Q$, $x\in U$. If the attributes in S can be divided into condition attribute set C and decision attribute set D, i.e. $Q=C\cup D$ and $C\cap D=\Phi$, the information system S is called a decision system or decision table.

Definition 2 Let $IND(P)$, $IND(Q)$ be indiscernible relations determined by attribute sets P, Q, the P positive region of Q, denoted $POS_{IND(P)}(IND(Q))$ is defined as follows:

$$POS_{IND(P)}(IND(Q)) = \bigcup_{X\in U/IND(Q)} IND(P)_{-}(X).$$

Definition 3 Let P, Q, R be an attribute set, we say R is a reduct of P relative to Q if and only if the following conditions are satisfied:

(1) $POS_{IND(R)}(IND(Q)) = POS_{IND(P)}(IND(Q))$;

(2) $\forall r\in R$ follows that

$$POS_{IND(R-\{r\})}(IND(Q)) \neq POS_{IND(R)}(IND(Q))$$

Definition 4 Let $L=(U, A\cup\{d\}, V, f)$ be a decision system, whose discernibility matrix $\boldsymbol{M}(U) = [M_A^d(i,j)]_{n\times n}$ is defined as:

$$M_A^d(i,j) = \begin{cases} \{a_k \mid a_k \in A \wedge a_k(x_i) \neq a_k(x_j)\}, & d(x_i) \neq d(x_j); \\ \Phi, & d(x_i) = d(x_j). \end{cases}$$

where $a_k(x_j)$ is the value of objects x_j on attribute a_k,

$d(x)$ is the value of object x on decision attribute d.

Write $\boldsymbol{M}(U)=[M_A^d(i,j)]_{n\times n}$ as a list $\{p_1,\ldots,p_t\}$. Each p_i is called a discernibility entry, and is usually written as $p_i=a_{i1}, \ldots, a_{im}$, where each a_{ik} corresponds to a condition attribute of the information system, $k=q,\ldots,m$; $i=1,\ldots,t$.

Furthermore, the discernibility matrix can be represented by the discernibility function f, conjunction normal form (CNF), i.e., $f=p_1\wedge\ldots\wedge p_t$, where each $p_i=a_{i1}\vee\ldots\vee a_{im}$ is called a clause, and each a_{ik} is called an atom. Note that the discernibility function contains only atoms, but not negations of atoms.

Although the discernibility matrix and discernibility function have different styles of expression, they are actually the same in nature.

Definition 5 Let h denote any Boolean CNF function of m Boolean variables $\{a_1^*,...,a_m^*\}$, composed of n Boolean sums $\{s_1,...,s_n\}$. Furthermore, let $w_{ij}^*\in\{0,1\}$ denote an indicator variable that states whether a_i^* occurs in s_j. $s_j=\sum_{i=1}^{m} w_{ij}^*\times a_i^*$, $h=\prod_{j=1}^{n} s_j$. We can interpret h as a bag or multiset $\boldsymbol{M}(h)=\{S_i \mid S_i=\{a_j\in A \mid a_j^* \text{ occurs in } s_i\}\}$. Because the discernibility function f is also a CNF Boolean function, so it has a multiset. Let $\boldsymbol{M}(f)$ denote the multiset of discernibility function f, $\boldsymbol{M}(f)=\{\{a_{11},\ldots,a_{1m}\},\ldots,\{a_{i1},\ldots,a_{im}\},\ldots,\{a_{t1},\ldots,a_{tm}\}\}$.

Definition 6 A hitting set of a given bag or multiset M of elements from 2^A is a set $B\subseteq A$ such that the intersection between B and every set in M is non-empty. The set $B\in HS(S)$ is a minimal hitting set of M if B ceases to be a hitting set if any of its elements are removed. Let $HS(M)$ and $MHS(M)$ denote the sets of hitting sets and minimal hitting sets, respectively,

$$HS(M)=\{B\subseteq A \mid B\cap S_i\neq\Phi \text{ for all } S_i \text{ in } M\}.$$

Proposition 1 For decision system $L=(U, A\cup\{d\}, V, f)$, g is its discernibility matrix, and $B\subseteq A$, $B\in RED(U, d)$ is equivalent to $B\in MHS(M(g))$ (Aleksander, 1999). So the rough set reduct computation can be viewed as a minimal hitting set problem.

Definition 7 A approximate hitting set is a set that hits "enough" elements of the bag or multiset M. The approximate hitting set provides an approximate solution to the hitting set problem. The set of ε-approximate hitting sets of the multiset M is denoted $AHS(M, \varepsilon)$:

$$AHS(M,\varepsilon)=\{B\subseteq A \mid \frac{|S_i \text{ in } M \mid S_i\cap B\neq\Phi|}{|M|}\geq\varepsilon\},$$

where the parameter ε controls the degree of approximation. The set is a minimal ε-approximate hitting set if it ceases to be so if any of its elements are removed. The set of all minimal ε-approximate hitting set is denoted $MAHS(M, \varepsilon)$.

Computing all elements of $MAHS(M, \varepsilon)$ is computationally intractable, and heuristics are needed (Wroblewski, 1995). In this paper, a heuristic rule based on the significance of attribute is applied to search for solutions.

Definition 8 The significance of attribute is defined as: $SGF(a, R, D)=p(a)$, $p(a)$ is the number of appearing times of attribute a in the remain part of the discernibility matrix which removes all the elements that have non-empty intersection with R.

Part B: Rough set classification algorithm

Our general scheme of rough set classification algorithm is presented in Fig.1.

(1) The raw input data set is transformed into a decision system which is subsequently split into two parts: the training dataset and the testing dataset. A classifier will be induced from the training dataset and applied to the testing dataset to obtain a performance estimate.

For the training dataset, we do the following steps (2), (3);

(2) If the decision system has real values attributes, the discretization strategies should be built to obtain a higher quality of classification rules. There are many discretizaion methods such as boo reasoning algorithm, semi-naïve algorithm etc. How

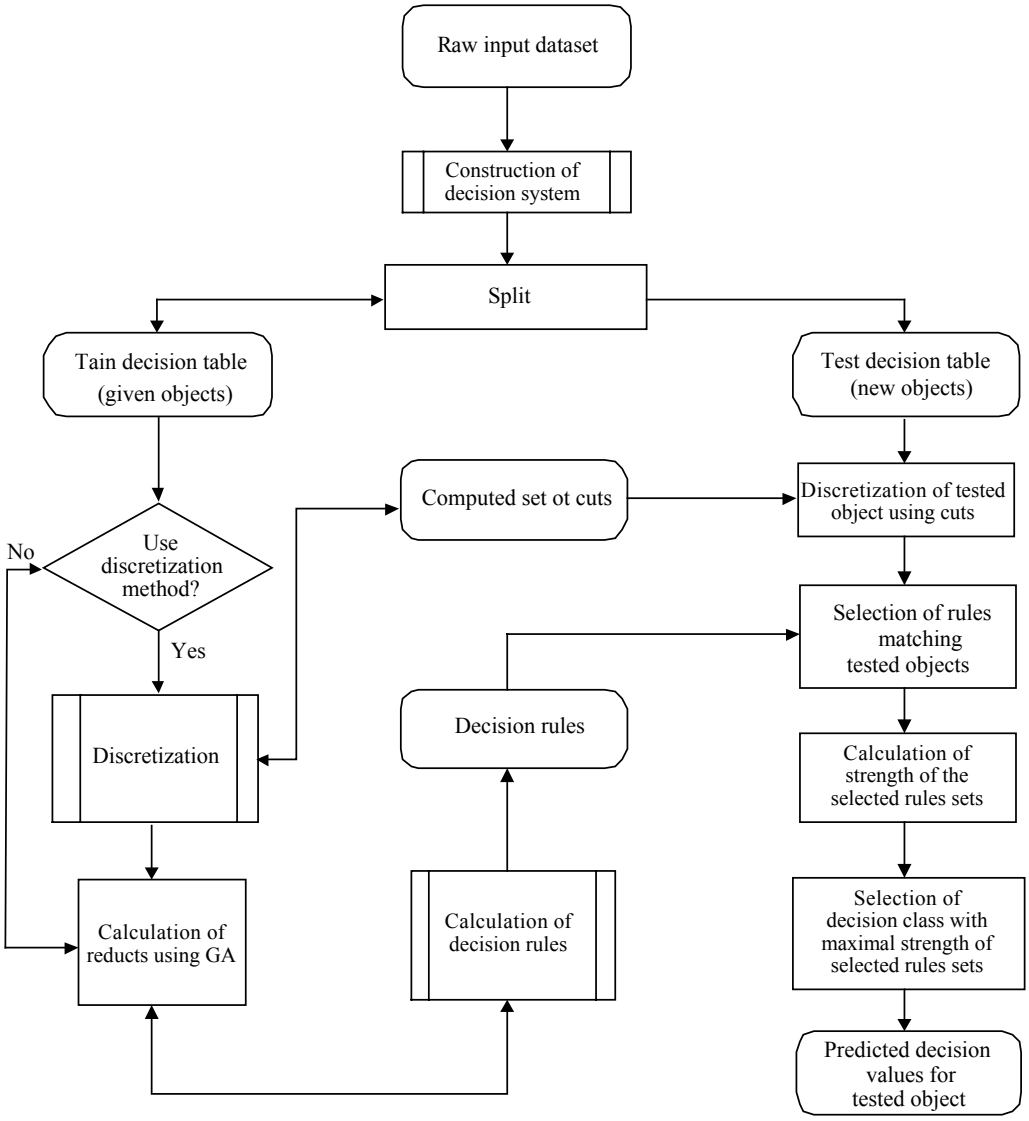

Fig.1 General scheme of rough set classification algorithm

to choose the suitable discretizaion methods is still a difficult question and some tests are needed. In our following experiment, Equal Interval Width discretizaion method is used. The Equal Interval Width discretization method divides the range of observed values for an attribute into k equal sized intervals, where $k>0$ is a user-supplied parameter. If an attribute a is observed to have values bounded by $a_{\min}$ and $a_{\max}$ then this method computes the interval width $width(k)=(a_{\max}-a_{\min})/k$ and constructs thresholds at $a_{\min}+i*width(k)$ where $i=1, ..., k-1$. The method is applied to each continuous attribute independently. Since this unsupervised method does not utilize decision values in setting partition boundaries, it is likely that classification information will be lost by binning as a result of combining values that are strongly associated with different classes into the same interval. But in our case this could make effective classification.

(3) The intrusion (decision) rules are created using the reducts computed by the attribute reduction algorithm as templates. There are many attribute reduction algorithms such as dynamic reduct (Bazan *et al.*, 1994) and RA-Order algorithm (Wang and Tao, 2003), etc. But, until now, the most effective algorithm for large decision system re-

duction computation in practice is genetic algorithm (Wroblewski, 1995), which is used by most of the rough set tools such as Rough Enough (Anders, 1997) and Rosetta (Aleksander, 1999). In this paper, we proposed a hybrid genetic algorithm based on the attribute significance heuristic rule to find minimal reducts. This hybrid genetic algorithm decreases the training time and makes the generated classifier more effective and it is adjusted to fit the intrusion detection environment. This hybrid genetic algorithm is the key sub-algorithm in our RSC Algorithm. In order to make it clear, we first introduce the general Genetic Algorithms (GAs) and their extension in Part C, then we describe the key sub-algorithm in detail in Part D.

For the testing dataset, the following step is done.

(4) The same cuts computed from training dataset discretizaion method are first used to discretize the new object dataset. Then the rules generated are used to match testing objects to compute the strength of the selected rule sets for any decision class. The new object will be assigned to the decision class with maximal strength of the selected rule set.

Part C: General genetic algorithms and their extension

GAs were formally introduced in the United States in the 1970s by John Holland at the University of Michigan (Goldberg, 1989). They have a solid basis in genetics and evolutionary biological systems. GAs comprise a kind of effective searching and optimizing technique and have been applied to various fields. In particular, GAs work very well on combinatorial problems such as reduct finding in rough set theory (Wroblewski, 1995).

A GA starts by generating a large set of possible solutions to a given problem (a solution to a problem corresponds to a genome or chromosome in genetics; a large set of possible solutions to a given problem corresponds to a population). It then evaluates each of those solutions, and decides on a "fitness level"("survival of the fittest" in the Darwinian principle of natural selection; the "fitness level" is represented by fitness function in GA) for each solution set. These solutions then breed new solutions ("breed" action is simulated by genetic operator, such as mutation, crossover and inversion, etc., in GA). The parent solutions that have better "fitness level" are more likely to reproduce (selection and reproduction strategy in GA), while those that have less "fitness level" are more unlikely to do so. In essence, solutions are evolved over time or generation (GA will iterate many times in control for optimization). This way GAs evolve the search space scope to a point where the solution can be found. Genetic algorithms can be incredibly efficient if programmed correctly.

The general algorithm for genetic algorithms includes the following steps:

Step 1. Generate an Initial Population

An initial population is created from a random selection of solutions.

Step 2. Evaluate Fitness

A value for fitness is assigned to each solution (chromosome) depending on how close it actually is to solving the problem (thus arriving to the answer of the desired problem). These "solutions" are not "answers" to the problem but are possible characteristics that the system would employ in order to reach the answer.

Step 3. Reproduce, Selection, Mutate and Crossover

Those chromosomes with a higher fitness value are more likely to reproduce offspring (which can mutate/inverse after reproduction). The offspring is a product of the father and mother, whose composition consists of a combination of genes from them (known as "crossing over").

Step 4. Control Next Generation

If the new generation contains a solution that produces an output that is close enough or equal to the desired answer then the problem has been solved. If this is not the case, then the new generation will go through the same process as their parents did. This will continue until a solution is reached. Then the algorithm is over.

We can extend the above general genetic algorithms to find rough set reducts quickly. The problems such as reduct finding are NP-hard, and there is no fast and reliable way to solve them in

deterministic way. Genetic algorithms are flexible and universal and can be used to solve these kinds of combinational problem. On the other hand, approximate but fast heuristics such as *SGF* approach for reduct finding (Wang and Tao, 2003) are designed and tuned up especially for those tasks, and are often more efficient than general genetic algorithm. Unfortunately, they are often suboptimal and cannot avoid local optima. Moreover, the deterministic algorithm such as reduct computation based on *SGF* often spend more time on computations and be no hope for improvement. The hybrid genetic algorithms (Wroblewski, 1995) which use nondeterministic, problem-oriented heuristics controlled by GAs can exploit the advantages of both genetic and heuristic algorithms. Different from the hybrid genetic algorithm by Wroblewski (1995), the heuristic method based on the significance of attribute $SGF(a, R, D)$ is introduced in our rough set classification algorithm by a new Bit-adapt operator. This Bit-adapt operator is our main extension of Step 3 in the above general genetic algorithm (see the following part and Fig.2 for details).

```
P←initializePopulation();
evaluate(P);
while(not terminate(P)) do
   Parents[1..3]←selectParents(P);
   Offspring[1]←CrossoverParents(Parents[1]);
   Offspring[2]←Mutation(Parents[2]);
   Offspring[3]←Inversion(Parents[3]);
   P←recombine (P, Offpring[1..3], Parents[1..3]);
   Bit-Adapt(P); //this operator implements an adaptation strategy which will be discussed in the following; after this operation, the attribute subset (represented by each chromosome) has the approximate classification ability of the whole condition attribute set;
   evaluate(P);
done
```

Fig.2 Pseudo-code for the hybrid genetic algorithm based SGF

Part D: Finding minimal reduct using hybrid genetic algorithm based on SGF

(1) Frame of hybrid Genetic Algorithm

As the above "Part A" said, finding the rough set minimal reduct is viewed as minimal hitting set problem. For the discretized decision system $L=(U, A\cup\{d\}, V, f)$, a multiset is constructed according to the above Definition 5, Part A. Subsequently, the hitting set of this multiset is computed using hybrid genetic algorithm.

As the above "Part C" said, when suitable local search techniques which make use of some heuristic information are introduced into simple genetic algorithms, the heuristic algorithms will maintain the capability of optimizing globally and can converge faster. Different from the hybrid genetic algorithm by Wroblewski (1995), the heuristic method based on the significance of attribute $SGF(a, R, D)$ defined in the Definition 8, Part A is introduced into the genetic algorithm for minimal approximate hitting set, i.e., approximate minimal relative reduct. The algorithm skeleton is shown in the following Fig.2.

In our algorithm, a new operator named Bit-adapt is constructed. This new operator operates on the whole population and can guarantee each chromosome converges into one of the hitting sets.

(2) Representation (Generation of the Initial Population)

For the minimal hitting set problem, a straightforward choice of population is a set P of elements from 2^A, encoded as bit-vectors, where each bit indicates the presence of a particular element in the set. For example, assume that we have 10 condition attributes $\{a_1, a_2, \ldots, a_{10}\}$ and we have a reduct candidate as $\{a_1, a_4, a_6, a_9\}$. Then the reduct candidate should be represented as: 1001010010.

(3) Function of fitness

According to the definition of relative reducts, we know that the fitness function depends on the assumption: the number of attributes (which we wish to keep as low as possible) and the decision ability (which we wish to keep as high as possible). Our fitness function for decision system $L=(U, A\cup\{d\}, V, f)$ is defined as follows: Let n denote the number of condition attributes, M the multiset of discernibility function of L and $B\subseteq A$,

$$f(B)=\frac{n-|B|}{n}+\min\{\varepsilon,\frac{|\{S \text{ in } M \mid S\cap B\neq\Phi\}|}{|M|}\}.$$

The first term rewards the shorter elements and the second tries to ensure that we reward sets that are hitting sets to guarantee the decision ability. The parameter ε controls the degree of approximation decision ability.

(4) Selection and recombination method

The selection and recombination operator are implemented with two steps:

Step 1: Calculate the fitness for each chromosome in the current generation t. Then according to the fitness for each chromosome, we use stochastic sampling method to select;

Step 2: Let *minsingle*(*Offspring*) be the worst individual in the new population, *minfit*(*Offspring*) be the corresponding fitness; Let *maxsingle*(*Parent*) be the best individual in the old population, *maxfit*(*Parent*) be the corresponding fitness. If *minfit*(*Offspring*)<*maxfit*(*Parent*), we replace *minsingel*(*Offspring*) with *maxsingle*(*Parent*).

(5) Crossover, Mutation and Inversion

We use classical, one-point crossover. Crossing-over process affects chromosome selected for reproduction with probability of P_c. In the mutation process, we first select a chromosome to be mutated with probability P_m and then choose a single gene of the chromosome randomly. Mutation of a single gene means replacement of "1" by "0" or "0" by "1". Suppose that chromosome $S_1=\{s_{11}, s_{12}, \ldots, s_{1r}, s_{1,r+l}, s_{1,r+l+1},\ldots, s_{1n}\}$, where r, l are random numbers. S_2 is the inversion of S_1: $S_2=\{s_{11}, s_{12}, \ldots, s_{1r}, s_{1,r+l}, s_{1,r+l+1},\ldots, s_{1n}\}$;

(6) Bit-Adaptation Strategy

We use this strategy as heuristic rule to make genetic algorithm converge faster. It is not time-consuming in computing. This operator operates on the whole population.

Let R be the attribute set represented by current chromosome. If R is not a hitting set (It is judged in the fitness function computation), then find an attribute a in $C–R$ which has the maximal value $SGF(a, R, D)=p(a)$. If there are several a_j, (i=1,2,…,m;) with the same maximal value, stochastically choose one attribute from them. Set the bit corresponding with a_j as "1".

When computing the fitness function of each individual, the multiset should be searched once; at the same time, the *SGF* values are computed in this same search procedure. So it will not remarkably increase the computation time of general GA with this "modify strategy". Our fitness function design can guarantee that the changed chromosome can converge faster and correctly.

EXPERIMENTS

In order to compare RSC algorithm with the classical learning algorithm for intrusion detection, we constructed intrusion detection systems using rough set classification (RSC) and support vector machines (SVM) and tested their performance on the 1999 KDD intrusion detection contest data set. All the two experiments (RSC based IDS and SVM based IDS) were done in the same Personal Computer (DELL Optiplex GX400 system) with 1.70 GHz Pentium IV CPU, Windows 2000 operating system and 128 M RAM. The compiler is Microsoft Visual C++ 6.0 and the program language is C and C++.

Part A: Development of RSC based IDS

The raw data from the KDD99 contest is first partitioned into three groups (input dataset): DoS attack detection dataset, Probe attack detection dataset, U2R&R2L attack detection dataset. For each input dataset, we construct a decision system using the following method: 1) for each attack detection dataset, different connection record feature set are selected as the condition attributes of the decision system. For Probe and DoS attack, intrinsic and traffic features are used; for U2R&R2L attack, intrinsic and content features used. 2) the label (normal+1, attack–1) variable of each record is used as the decision attribute of the decision system. 3) a connection record data point is used as an object in the decision system. Each of the constructed decision systems will be processed by RSC subsequently.

DoS attack detection dataset consists of the set of 5330 data points: 4264 for training, 1066 for testing. Data points contain 3206 actual DoS attack connection records and 2124 normal usage pattern

data points. These data points are used for training using RSC algorithm. The generated rules are used to predict the tested objects.

Probe attack detection dataset consists of the set of 2434 data points: 1947 for training, 487 for testing. Data points contain 310 actual Probe attacks connection records and 2124 normal usage pattern data points. Data points are used for training using RSC algorithm. The generated rules are used to predict the tested objects.

U2R&R2L attack intrusion dataset consists of the set of 2214 data points: 1171 for training, 1043 for testing. Data points contain 90 actual U2R&R2L attacks connection records and 2124 normal usage pattern data points. Data points are used for training using the RSC algorithm. The generated rules are used to predict the tested objects.

The experiments results are shown in Table 1, where the training time units format is minutes:seconds. Training time 1 denotes the training time without "Bit-Adaptation strategy" in the genetic algorithm; training time 2 denotes the training time with "Bit-Adaptation strategy". RSC-ε refers to the parameter ε used in the reduct computation. ε=1 means it is the accurately computed hitting set without approximation.

From the above table, our proposed "Bit-Adaptation strategy" can decrease the training time.

Part B: Development of SVM based IDS

All the three input dataset construction and the experiment platform are the same as RSC IDS for convenient comparison. The only difference between the two systems are Step 4) in the SYSTEM MODEL. In RSC based IDS, the learning algorithm is RSC; but in SVM based IDS, the learning algorithm is SVM. The other steps in the SYSTEM MODEL are the same. In addition, the SVM based IDS experiments described below use the freeware package LIBSVM (Chang and Lin, 2003).

Each of the above three categories of input dataset is preprocessed into the LIBSVM input format and then is scaled using linear scaling. Furthermore, the SVMs are trained using the RBF (radial bias function) kernel option with different parameter (C, cost; g, gamma) for each category. Each data point is located in n-dimensional space, with each dimension corresponding to a feature of the data point.

The experiment results are shown in Table 2. Our results are similar to those in (Srinivas and Sung, 2002).

Part C: Discussion and comparison of SVM IDS and RSC IDS

Table 3 shows the optimal experiment results of RSC and SVM algorithms.

From Table 3, we can conclude that RSC algorithm has detection performance level compatible with that of the SVM algorithm in terms of Probe and DoS attack detection (all above 99%). But for U2R&R2L attack detection, RSC algorithm is worse than SVM algorithm. The reason is that RSC algorithm can get good performance when the samples are enough while it performs a little worse for small attack sample case (In the DARPA dataset, U2R&R2L attack samples are low but DoS and Probe attack samples are enough). In contrast, SVM is a good tool that performs well for both small and enough sample attack cases.

Furthermore, the detection rules generated by

Table 1 Experiment results of RSC based IDS

Category	RSC-ε	Detection rate	Misclassification rate	Training time 1 (min:sec)	Training time 2 (min:sec)
Probe attack	ε=0.9	0.9968	0.0	1:39	1:29
	ε=1	0.9968	0.0	1:10	1:01
DoS attack	ε=0.9	0.9298	0.0	4:30	3:59
	ε=1	0.9975	0.0005	3:29	3:07
U2R&R2L attack	ε=0.9	0.6875	0.0	0:12	0:12
	ε=1	0.7368	0.0	0:10	0:10

Table 2 Test results of SVM based IDS (Using RBF kernel function with different parameters)

Category	Training parameter	Support vector number	Iteration number	Detection rate	Training time (s)	Testing time (s)
Probe attack	C=1, g=0.033	56	42	99.5892%	0.82	0.40
	C=1.5, g=0.033	44	53	99.5892%	0.82	0.39
	C=10, g=0.1	30	67	99.9178%	0.91	0.41
	C=50, g=1	148	145	99.7535%	1.03	0.43
DoS attack	C=1, g=0.033	145	102	99.2495%	1.10	0.39
	C=1.5, g=0.033	123	137	96.1764%	1.10	0.39
	C=10, g=0.1	56	151	98.1614%	1.20	0.40
	C=50, g=1	209	204	98.9686%	2.33	0.56
U2R&R2L attack	C=1, g=0.033	25	26	99.5331%	0.13	0.11
	C=1.5, g=0.033	22	27	99.5331%	0.13	0.11
	C=10, g=0.1	29	117	99.6265%	0.15	0.12
	C=50, g=1	34	211	99.1597%	0.17	0.13

Table 3 Comparison of SVM and RSC algorithm

Category	SVM (c=1, g=0.033)		RSC (ε=1)	
	Detection rate	Misclassification rate	Detection rate	Misclassification rate
Probe	99.59%	0.01643	99.68%	0
Dos	99.25%	0.03	99.75%	0.0005
U2R&R2L	99.53%	0.4424	73.68%	0

the RSC algorithm, different from SVM, has the explainable "IF-THEN" format. For instance, one of the Probe attack detection rules is: "duration([1, 2080]) AND dst_host_srv_rerror_rate([99, *]) => decsion(−1)". Its meaning is that "if the rate of connections that have 'REJ' errors of the same service of 100 connections to the same host (the definition of 'dst_host_srv_rerror_rate' (Wenke, 1999)) is above 99%, then it is a probe attack". Since the packets of probe attack often access the unopened service, it can be concluded that the rule is evident. With this learned knowledge, we can improve the design of the probe detector. This advantage, together with its high detection performance for some attacks, makes the RSC algorithm very valuable in practical intrusion detector design. On the other hand, SVM had been criticized for the difficulties in model explanations.

Another difference between RSC and SVM is the feature ranking for IDS. RSC accomplished the feature reduct computation once before detection rules generation. In fact, the reduct computation corresponds to feature ranking for IDS. So, feature ranking is performed only once for RSC. However, feature ranking using SVM needs many times iterations. The SVM based feature ranking approach (Srinivas and Sung, 2002) is: one input feature is deleted from the data at a time, the resultant data set is used for the training and testing of the classifier; then the classifier's performance is compared to that of the original classifier (based on all features) in terms of relevant performance criteria; finally, the importance of the feature is ranked according to a set of rules based on the performance comparison. Compared with SVM, RSC feature ranking is simpler and faster.

On the other hand, SVM has good scalability. And the training and running time for SVM is significantly shorter (e.g. 1.03 s vs 1.39 min for Probe attack). Compared with the SVM, the training time of RSC is longer. Although we use the heuristic rule to accelerate the convergence speed of the reduct

computation and decrease the training time of RSC, the training time of RSC is still long and needs further improvement. However, the running time of RSC is notably short since it just needs judgment of some conditions.

CONCLUSION

It is very valuable to get both high detection rate and explainable rules since this can improve our knowledge about the nature of the intrusion. In this paper we use rough set classification (RSC) for intrusion detection system (IDS) feature ranking and intrusion detection rules generation. Intrusion detection using RSC can yield both explainable detection rules and high detection rate for some attacks. And feature ranking using RSC for IDS is simple and fast.

In addition, we proposed a hybrid genetic algorithm based on the attribute significance to compute the rough set reduct and accelerate the convergence speed and decrease the training time of RSC. But for the real-time IDS, the training time of RSC is still long and needs further improvement.

ACKNOWLEDGMENT

We would like to acknowledge many valuable conversations with Professor Sun Shi-yin, IEEE senior member, from Shanghai Jiaotong University, Chen-hui, Research Associate from Hong Kong Polytechnic University, Hong Kong, and Liu Dong-xi, Research Fellow from the Department of Computer Science School of Computing, National University of Singapore, Singapore.

References

Aleksander, Ø., 1999. Discernibility and Rough Sets in Medicine: Tools and Applications. PhD Dissertation, http://www.idi.ntnu.no/~aleks/thesis.

Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J., Stoner, E., 2000. State of the Practice of Intrusion Detection. Technical Report, http://www.sei.cmu.edu/pub/.

Anders, T.B., 1997. Rough Enough–A System Supporting the Rough Sets Approach. Sixth Scandinavian Conference on Artificial Intelligence SCAI'97.

Bazan, J.G., Skowron, A., Synak, P., 1994. Dynamic Reducts as A Tool for Extracting Laws from Decision Tables. Proceedings of ISMIS'94. Lecture Notes in Artificial Intelligence 869. Springer-Verlag, Berlin, p.346-355.

Chang, C., Lin, J., 2003. LIBSVM, A Library for Support Vector Machines. http: //www.csie.ntu.edu.tw/~cjlin/libsvm/.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, M.A.

James, C., 1998. The Application of Artificial Neural Networks to Misuse Detection: Initial Results. RAID98, Louvain-la-Neuve, Belgium, p.14-16.

KDD, 1999. http://kdd.ics.uci.edu/databases/kddcup99/task.html.

Pawlak, Z., 1982. Rough sets. *International Journal of Computer and Information Sciences*, **11**:341-356.

Srinivas, M., Sung, A., 2002. Feature Ranking and Selection for Intrusion Detection. Proceedings of the International Conference on Information and Knowledge Engineering.

Wang, G.Y., eds, 2001. Rough Set Theory and Knowledge Acquistion. Xi'an Jiaotong University Press, Xi'an (in Chinese).

Wang, J., Tao, Q., 2003. Rough Set Theory and Statistical learning Theory. *In*: Lu, R.Z., ed., Knowledge Science and Computing Science. Tsinghua University Press, Beijing, p.49 (in Chinese).

Wenke, L., 1999. A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems. PhD dissertation, http://www.cc.gatech.edu/~wenke/.

Wroblewski, J., 1995. Finding Minimal Reducts Using Genetic Algorithms. Proc. of the second Annual Join Conference on Information Sciences. Wrightsvillle Beachm, NC, p.186-189.